Chapter 1

On the Kolmogorov-Chaitin Complexity for short sequences

Jean-Paul Delahaye* and Hector Zenil†

Laboratoire d'Informatique Fondamentale de Lille
Centre National de la Recherche Scientifique (CNRS)
Université des Sciences et Technologies de Lille

A drawback to Kolmogorov-Chaitin complexity (K) is that it is uncomputable in general, and that limits its range of applicability. Moreover when strings are short, the dependence of K on a particular universal Turing machine U can be arbitrary. In practice one can approximate it by computable compression methods. However, such compression methods do not provide a good approximation for short sequences — shorter for example than typical compiler lengths. In this paper we will suggest an empirical approach to overcome this difficulty and to obtain a stable definition of the Kolmogorov-Chaitin complexity for short sequences. Additionally, a correlation in terms of distribution frequencies was found across the output of several systems including abstract devices as cellular automata and Turing machines, as well as real-world data sources such as images and human DNA fragments. This could suggest that all them stand a single distribution in accordance with algorithmic probability.

1.1. Introduction

Among the several new ideas and contributions made by Gregory Chaitin to mathematics is his strong belief that mathematicians should transcend the millenary theorem-proof paradigm in favor of a quasi-empirical method based on current and unprecedented access to computational resources.[3] In accordance with that dictum, we present in this paper an experimental approach for defining and measuring the Kolmogorov-Chaitin complexity, a problem which is known to be quite challenging for short sequences — shorter for example than typical compiler lengths.

*delahaye@lifl.fr
†hector.zenil-chavez@malix.univ-paris1.fr

The Kolmogorov-Chaitin complexity (or algorithmic complexity) of a string s is the length of its shortest description p on a universal Turing machine U, formally $K(s) = min\{l(p) : U(p) = s\}$. An important property of K is that it is nearly independent of the choice of U. The major drawback of K, as measure, is its uncomputability. So in practical applications it must always be approximated by compression algorithms. Moreover, when strings are short, the dependence of K on a particular universal Turing machine U can be arbitrary. In this paper we will suggest an empirical approach to overcome this difficulty and to obtain a stable definition of the Kolmogorov-Chaitin complexity for short sequences.

Using Turing's model of universal computation, Solomonoff[9,10] produced a universal prior distribution deeply related to the Kolmogorov-Chaitin complexity. This work was later generalized and extended by a number of researchers, in particular Leonid A. Levin[7] who formalized the initial intuitive approach in a rigorous mathematical setting and supplied the basic theorems of the field. There now exist a number of strongly related universal prior distributions; however, this article will describe only the discrete universal probability. Further details on the algorithmic complexity and the universal distribution are given in the works of Chaitin,[1,2] Calude,[4] Li and Vitányi.[5,6]

Algorithmic probability is the probability $m(s)$ that a universal Turing machine U produces the string s when provided with an arbitrary input tape. $m(s)$ can be used as a universal sequence predictor that outperforms (in a certain sense) all other predictors.[5] Its main drawback is that it is not computable either and thus can only be approximated in practice. It is easy to see that this distribution is strongly related to Kolmogorov complexity and that once $m(s)$ is determined so is $K(s)$ since the formula $m(s)$ can be written in terms of K as follows $m(s) \approx 1/2^{K(s)}$.

Turing machines are extremely basic abstract symbol-manipulating devices able of universal computation. A Turing machine can run forever, enter a loop, or reach a particular state or condition (reaching certain head position or producing certain output) at which it is prescribed to halt. The most commonly considered Turing machines have either 0 or 1 halting states. A Turing machine with 0 halting states is just a Turing machine which can run forever despite the reached condition even if it is staled. In such a case the halting state is determined after an arbitrary number of steps.

Other very simple computational devices are called cellular automata (CA). A cellular automaton is a collection of cells on a grid that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells. The rules are applied iteratively for as many time steps as desired.

1.2. The procedure

The whole experiment was performed for two type of systems: abstract automaton such as Turing machines and cellular automata, and real-world repositories containing information from the physical world, such as images and DNA sequences.

For all cases n will denote the size of the string to look for and compare with at each stage. Usually a string will be denoted by an $s \in \{0,1\}^*$. In the case of abstract automata we look at their output (binary strings) after letting them run certain defined steps. For the case of real-world repositories we transform their content to binary strings and in both cases those strings were partitioned according to the given length n, then they were regrouped by frequency. Such regrouping should yield a classification that should follow a distribution according to their Kolmogorov-Chaitin complexity as predicted by algorithmic probability and as described in the previous section.

By means of experiments with abstract computational devices such as Turing machines and cellular automata, we claim that it might be possible to overcome the problem of defining and measuring the Kolmogorov-Chaitin complexity of short sequences. Our proposal consists of measuring the Kolmogorov-Chaitin complexity by reconstructing it from scratch.

1.2.1. *Abstract computing machines*

Our experiment proceeded as follows: we looked at the output of a complete class of abstract devices, following an enumeration proposed by Stephen Wolfram.[11] For the case of Turing machines (TM) we performed experiments over the whole classes of (a) $2-state$ $2-symbol$ and (b) $3-state$ $2-symbol$ Turing machines, henceforth denoted as $TM(2,2)$ and $TM(3,2)$. For (a) it turns out that there are 4096 different Turing machines according to the formula $(2sk)^{sk}$ derived from the traditional $5-tuplet$ description of a Turing machine: $d(s_{\{1,2\}}, k_{\{1,2\}}) \rightarrow (s_{\{1,2\}}, k_{\{1,2\}}, \{1,-1\})$ where $s_{\{1,2\}}$ are the two possible states, $k_{\{1,2\}}$ are the two possible symbols and the last entry {1,-1} denotes the movement of the head either to the right or to the left. From which it follows that all possible $3-state$ $2-symbol$ Turing machines are given by $(2 \cdot 3 \cdot 2)^{2 \cdot 3} = 2985984$. So for (b) we proceeded by statistical methods, taking samples of size 5000, 10000, 20000 and 100000 Turing machines uniformly distributed over the complete class of $TM(3, 2)$. We then let them run 30, 100 and 500 steps each and we proceeded to feed each one with (1) a (pseudo) random (one per TM) input and (2) with a regular input.

In the second part of the experiment with abstract devices, we proceeded exactly in the same fashion for cellular automata with (1) nearest-neighbor and (2) two neighbors on the left and one on the right, both taking a single 1 on a background of 0s, henceforth denoted by $CA(t, c)$[a] where t and c are the neighbor cells in question, to the left and to the right respectively. We took the enumeration scheme developed by Stephen Wolfram[11] which consists of 256 nearest-neighbor cellular automata (ECA) and 65536 $CA(2, 1)$.

An important result was that the distribution frequencies in each case were very stable under the variations described before, allowing to define a *natural* distribution $m(s)$.

We then looked at the frequency distribution of the outputs of both classes TM and CA[b], (including ECA) performing experiments modifying several parameters: the number of steps, the length of strings, (pseudo) random vs. regular inputs, and the sampling sizes.

For each Turing machine output $s(TM(n), m)$ or cellular automata output $s(CA(n), m)$ we took all the strings of a fixed length k produced by either the n-th Turing machine or the n-th cellular automaton according to the enumerations of each one. Since sentences over any finite alphabet are encodable as bitstrings, without loss of generality we focus on the binary alphabet $\{0, 1\}$ to which all s belong. $s(TM(n), m)$ is what is written on the tape of the $TM(n)$ Turing machine after the first m steps taking the cells that were reached by the head. $s(CA(n), m)$ is the output of the cellular automaton $CA(n)$ taking the cells that could be affected over the course of m steps. If $s(X(n), m) = (s_1, \dots, s_u)$, we considered $(s_1, \dots, s_k), (s_2, \dots, s_{k+1}), \dots, (s_{u-k+1}, \dots, s_u)$, i.e. all the $u - k + 1$ k-tuples, with X either TM, ECA or CA[c].

By analyzing the diagram it can be deduced that the output frequency

[a] A better notation is the $3 - \textit{tuplet}\ CA(t, c, j)$ with j indicating the number of symbols, but because we are only considering $2 - \textit{symbol}$ cellular automata we can take it for granted and avoid that complication.

[b] Both enumeration schemes are implemented in Mathematica calling the functions CelullarAutomaton and TuringMachine, the latter implemented in Mathematica version 6.0

[c] It can be seen as a "partition" with an offset one.

distribution of each of the independent systems of computation follow a complexity distribution. We conjecture that these systems of computation and others of equivalent computational power converge toward a single distribution when bigger samples are taken by allowing a greater number of steps and bigger classes containing more and increasingly sophisticated computational devices. Such distributions should then match the value of $m(s)$ and therefore $K(s)$ by means of the convergence of what we call their experimental counterparts $m_e(s)$ and $K_e(s)$. If our method succeeds as we claim it could be possible to give for the first time a stable definition of the Kolmogorov-Chaitin complexity for short sequences independent of any constant.

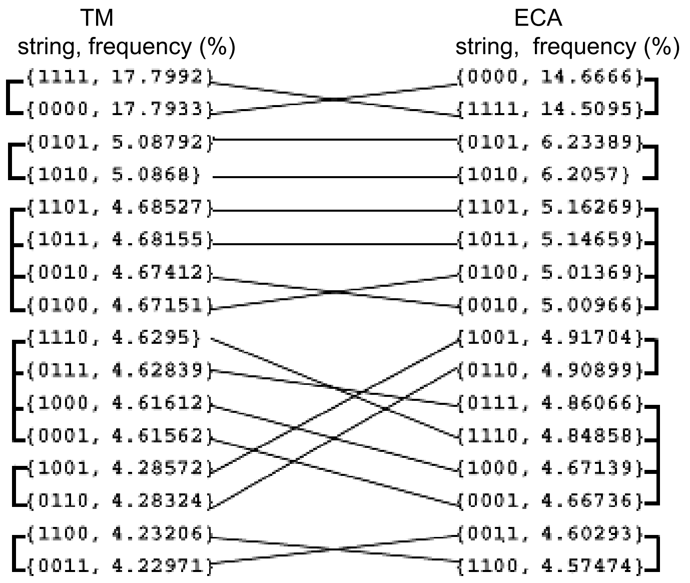

Fig. 1.1. The above diagram shows the convergence of the distribution frequency of the outputs of TM and ECA for $k = 4$. Matching strings are linked by a line. As one can observe, in spite of certain crossings, TM and ECA are strongly correlated and both successfully group equivalent output strings by frequency according to basic symmetries that preserve their Kolmogorov-Chaitin complexity; such as reversion (re), complementation (co) and the composition of (re) and (co). Moreover, by taking the six groups — marked with brackets — the distribution frequencies only differ by one.

Once verified that the generated distributions group the strings by the symmetries that preserve their Kolmogorov-Chaitin complexity (reversion (*re*), complementation (*co*) and the composition of (*re*) and (*co*)) and because, in a second stage, we were interested in counting and comparing the distributions between groups, a general theorem that allows to count the number of discrete combinatorial objects of a given type as a function of their symmetrical cases was applied. By using the Burnside's lemma (sometimes also called Pólya enumeration theorem) we simplified the total number of examined strings by grouping them into the representative instances of each complexity class[d] defined by the symmetrical cases that preserved the complexity of the original strings. Such a reduction[e] was given by the following formula according to the Burnside's lemma:

$$(2^n + 2^{n/2} + 2^{n/2})/4, \text{ for } n \text{ odd}$$
$$(2^n + 2^{(n+1)/2})/4 \text{ otherwise.}$$

As a consequence of having applied the Burnside's lemma, grouping the strings for simplification purposes, we had at the end to divide the frequency results by 2 or 4[f] according to the following formula:

[d]By instance, the string 0001 have the same Kolmogorov-Chaitin complexity than 0111, 1000 and 1110 which are strings obtained by applying any or a composition of reversion and complementation (denoted by t for transformation) to the original string.

[e]Following the Burnside's lemma the output was therefore reduced to the set of strings $s_1, s_2, \ldots, s_n \in S$ with $n \in N$ such that $\forall i, j\ t(s_i) \neq t(s_j)$ with $i \neq j$, $\{i, j\} \leq n$ and t. In such a way that no string $s_i \in S$ could be obtained from another $s_j \in S$ by using t. There are 2^n different binary strings of length n and 4 possible transformations to have in consideration for applying Burnside's lemma that preserve complexity:

(1) *id*, the identity symmetry, $id(s) = s$.

(2) *sy*, the reversion symmetry given by: If $s = d_1 d_2, \ldots d_n$, $sy(s) = d_n d_2, \ldots d_1$.

(3) *co*, the complementation symmetry given by $co(s) = mod(d_i + 1, 2)$.

(4) *syco*, the symmetry or reversion followed or preceded by complementation, i.e. any of the composition $sy(co)$ or $co(sy)$ which we are going to denote just as *syco*.

The number of cases were therefore reduced by applying the Burnside's lemma as follows: the number of invariant strings under *id* is 2^n. The number of invariant strings under *sy* is $2^{n/2}$, if n is even, and $2^{(n+1)/2}$ if n odd. The number of invariant strings under *co* is zero. The number of invariant strings under *syco* is $2^{n/2}$ if n is even, or zero if it is odd. Therefore, the number of different strings following the Burnside's lemma is given by:

[f]For example, the string $s_1 = 0000$ for $n = 4$ was grouped with the string $s_2 = 1111$ because by Burnside's lemma one of them suffices as a representative instance of the class of strings with their same algorithmic complexity $\{s_1, s_2\} = \{0000, 1111\}$. Another example of a symmetrical string $s = 0011$ with $\{s_1, s_2\} = \{0011, 1100\}$. By contrast, the string 0100 was grouped with four other strings since $\{s_1, s_2, s_3, s_4\} = \{0100, 0010, 1101, 1011\}$ had the same algorithmic complexity.

$$f_r(s)/|\bigcup(sy(s), co(s), syco(s))|$$

were f_r represents the frequency of the string s and the denominator is the cardinality of the union set of the equivalent strings under reversion, complementation, and composition of the string s.

1.2.2. ***Real-world data sources***

On the other hand, we were interested in comparing all those output strings from abstract machines studied above with repositories containing or capturing a part of the world around such as arbitrary images (IMG) and Deoxyribonucleic acid (DNA).

1.2.2.1. *B&W Images*

Because we wanted to simplify the experiment comparing strings of the same type, all strings from all systems involved were chosen or transformed into binary sequences. All images (IMG) were transformed into black-and-white images (0 for white and 1 for black[g]), and all automata experiments were also designed in binary (black-and-white cellular automata and $2-symbol$ Turing machines). This choice does not imply that the experiment cannot be performed comparing strings in other bases[h] using richer automata (e.g. using more colors in the case of cellular automata or extending the alphabet for the Turing machines allowing more symbols) and by taking color images. However, in such a case, the number of different strings and their algorithmic complexity would have complicated the experiment without (possibly) producing any different or further results.

Because we wanted to take a sample of the structured world around, and in order to compare it with the abstract computing devices studied before (cellular automata and Turing machines), we proceeded as follows: we took a representative set of arbitrary images or pictures. For our experiment, we took one hundred pictures from the web[i], by instance from the web[j].

[g]It is true that the nature of the images as digital repositories does not guarantee that they be a full description of a part of a physical *reality* but in any case the digital image is a limited but legitimate sample of such *reality*.

[h]Our approach also applies to sets of mixed strings of different lengths. However the approach followed in this paper only included comparisons between strings of the same size.

[i]The experiment was made three times for triple-checking with three different sets of 100 different images each one. All them produced the same result.

[j]From http://www.flickr.com, all of them public or under the Creative Commons License. A zipped file containing both color and B&W images is available at:

We converted all images into black-and-white with a threshold of 50, which means that the full color images were converted to high-contrast, black-and-white images as follows: all pixels lighter than the threshold were converted to white and all pixels darker were converted to black. Put another way, if the color entry of a pixel was x, x was sent to one or zero if it was darker or lighter than the threshold. The result was a balanced black-and-white image composed by a binary matrix. Each binary matrix row for each image was then partitioned into strings of length n with offset one. All strings were then reduced according to Burnside's lemma and grouped by frequency. The frequency was divided according to what we described before taking into consideration the result of applying the lemma.

1.2.2.2. *DNA analysis*

Another rich source of information to compare was the Deoxyribonucleic acid (DNA). The experiment with DNA was performed as follows: we took a sample of DNA. In our case the complete homo sapiens chromosome 1[k]. We transformed the data into a binary string. There are two possible encoding transformations for translating a DNA sequence into a binary string using a single bit for each letter (both yield to different classifications due to an asymmetry):

(1) G $\rightarrow$ 1, T $\rightarrow$ 1, C $\rightarrow$ 0, A $\rightarrow$ 0
(2) G $\rightarrow$ 1, T $\rightarrow$ 0, C $\rightarrow$ 1, A $\rightarrow$ 0

We partitioned the strings into parts of length n with offset one. All strings were then reduced according to Burnside's lemma and grouped by frequency. The frequency was divided according to what we described before taking into consideration the result of applying the lemma.

This is how all the classification arrays for $n = 5$ look like, including both types of Turing machines (TM and TMR), both types of Elementary Cellular Automata (ECA and ECAR), the B&W images (IMG) and the

ftp://ftp.mathrix.org/zenil/universaldistribution user:universalmathrix.org and password: distribution. The file is called Images.zip and ImagesBN.zip for the full-color images and black-and-white images respectively. By the name of the images it is possible to trace them to the original source in the above website. All the source code in PDF and in Mathematica notebooks, and further and detailed explanations of the whole experiment is available online at the same location zipped in a file called "ExperimentXXXXXX.zip" where "XXXXXX" is the date of the last build. At least Mathematica version 6.0 is required to perform the complete experiment.

[k]ref—NT_077402.1—Hs1_77451 1167280 Homo sapiens chromosome 1 genomic contig, reference assembly: http://www.ncbi.nlm.nih.gov

human DNA fragment (here it was taken the first possible encoding denoted by DNA1):

TM	n=5	TMR	n=5	ECA	n=5
string	frequency	string	frequency	string	frequency
00000	91.3356	00000	56.0163	00000	62.4994
01010	7.65965	00100	6.45114	01010	21.3888
01111	0.568328	01000	6.0789	00100	3.89111
00111	0.254498	01111	5.70299	01000	3.2993
01000	0.064866	01010	4.81349	01110	2.93436
01011	0.046982	01011	4.3669	01111	1.66198
00100	0.044853	01100	4.36167	01101	1.57321
01101	0.022994	00111	4.10775	00111	1.0899
01100	0.002271	01110	4.08366	01011	0.976476
01110	0.0	01101	4.01717	01100	0.685506
ECAR	n=5	IMG	n=5	DNA1	n=5
string	frequency	string	frequency	string	frequency
00000	39.7849	00000	90.8222	00000	29.8133
01010	9.67742	01111	2.66327	01111	11.5772
01000	7.52688	00111	2.05495	00100	10.8819
01101	6.98925	00100	1.18046	01000	10.0047
01100	6.98925	01000	1.14837	00111	7.9888
00100	6.98925	01100	0.706686	01100	7.41251
01111	5.91398	01110	0.541776	01110	6.22025
01110	5.91398	01101	0.363852	01101	5.86094
00111	5.64516	01011	0.331501	01011	5.59263
01011	4.56989	01010	0.186917	01010	4.64769

1.1.1 All classification arrays for $n = 5$.

1.3. Classification comparisons by the application of two metrics

Because we wanted to compare all the classifications that we obtained after the experiment in order to find some possible correlations between them we envisaged two different measures for calculating the distance between two classification arrays. In general two kinds of comparisons are possible: one at the level of the string ordering without taking into account specific frequency values, i.e. only the raw classification mattered, and one at a refined level taking into account the values of the frequencies of each string, i.e. their probability of appearance. Certainly there are other possible statistical methods for measuring the raw and probability distance between classification arrays of this type and we will explore them in the future.

1.3.1. *Raw classification distance*

Let $C_1 = \{i(1), i(2), \ldots, i(n)\}$ and $C_2 = \{j(1), j(2), \ldots, j(n)\}$ be two different classifications. Because C_1 and C_2 have the same length n and they contain the same strings but (possibly) in a different order, (which is precisely what we want measure), it is clear that C_2 is some permutation of the elements of C_1 which means that the distance between C_1 and C_2, denoted by $d(C_1, C_2)$ will be given by the following formula:

$$\begin{aligned} d(C_1, C_2) &= |i(1) - j(1)| + |i(2) - j(2)| + \cdots + |i(n) - j(n)| \\ &= \Sigma_{m=1}^{n} |i(m) - j(m)| \end{aligned} \tag{1.1}$$

So if two classification arrays differ by two, that would mean that a string appeared one place shifted upper or lower in the classification, or in other words that one would need to exchange one string for an adjacent one in order to get a classification match with the other. Note that the smallest possible distance after zero is two because if a string differs by one place, then the other differs also by one, and that makes two according to our distance formula. So any distance value should be divided by two if one wants to have an idea in these terms.

This is how a typical raw distance comparison looks like. Example for $n = 5$:

IMG	00000	01111	00111	00100	01000	01100	01110	01101	01011	01010
DNA	00000	01111	00100	01000	00111	01100	01110	01101	01011	01010

1.1.2 Table showing a typical comparison between two classification arrays disregarding the string frequencies. This case is one of the best correlated examples between images and the human being first chromosome DNA (first encoding).

According to our definition the raw distance between them is given by:

$$\begin{aligned} d(IMG, DNA) &= |i(1) - j(1)| + |i(2) - j(2)| + \cdots + |i(10) - j(10)| \\ &= |3 - 4| + |4 - 5| + |5 - 3| \\ &= 1 + 1 + 2 \\ &= 4 \end{aligned} \tag{1.2}$$

1.3.1.1. *Average raw distance*

Once calculated the distance between two classification arrays, namely C_1 and C_2, we wanted to compare it with an average case to determine the possible correlation between two classifications. To size the order of magnitude between distances we calculated the average case between randomly sorted classification arrays, which would be statistically the worst case meaning that they were not correlated. It follows that if $d(C_1, C_2) < D$ then C_1 and C_2 would be correlated.

1.3.2. *Probability distance*

The probability distance compares the distance according to the strings frequency. Denoted by d_{pr} the probability distance is given by the following formula:

$$d_{pr}(C_1, C_2) = (\Sigma_{s=00\ldots0}^{11\ldots1} |pr_1(s) - pr_2(s)|)/2^n \tag{1.3}$$

With C_1 and C_2 including all the 2^n strings s before applying the Burnside's lemma and pr_i the probability of s in C_i. We decided to divide by 2^n in order to obtain a result which would be proportional to the size of the set of all strings and thus permit the comparison between the distances involving different values of n.

This is how a typical probability distance comparison looks like for the same example used for the raw distance:

According to our definition the probability distance between the classifications in the table 1.1.3 is:

$$d_{pr}(C_1, C_2) = (\Sigma_{s=00\ldots0}^{11\ldots1} |pr_1(s) - pr_2(s)|)/2^n = 4.4 \tag{1.4}$$

IMG							
$0000, 45.6$	$0001, 0.7$	$0010, 0.3$	$0011, 1.3$	$0100, 0.3$	$0101, 0.2$	$0110, 0.5$	$0111, 0.7$
$1000, 0.7$	$1001, 0.5$	$1010, 0.2$	$1011, 0.3$	$1100, 1.3$	$1101, 0.3$	$1110, 0.7$	$1111, 45.6$
DNA							
$0000, 28.0$	$0001, 2.3$	$0010, 2.4$	$0011, 3.7$	$0100, 2.4$	$0101, 4.7$	$0110, 3.9$	$0111, 2.3$
$1000, 2.3$	$1001, 3.9$	$1010, 4.7$	$1011, 2.4$	$1100, 3.7$	$1101, 2.4$	$1110, 2.3$	$1111, 28.0$

1.1.3 Table showing a typical comparison between two classification arrays for $n = 4$ taking into account the frequencies of appearance of all $2^4 = 16$ strings. In this case for images (top) and DNA (bottom). Each cell shows the string followed by its frequency percentages.

1.3.2.1. *Average probability distance*

Based on the logic given in section 1.3.1.1, we calculated the average probability distance, denoted by the letters PR, between two arbitrary randomly generated classification arrays. In this case, because the distance takes into account the probability of appearance of each string it follows that assigning the same probability for each string in such a way that the average case is that one assigning the same probability to each of the strings as follows:

$$pr(s_i) = 1/2^n \text{ for all } i \in \{1, \ldots, n\} \tag{1.5}$$

Therefore, the average probability distance between two random arrays turns out to be the equi-distribution of the total probability divided by the number of total strings.

So if $n = 4$, a random array C_r shall have $1/2^4 = 1/16$ as average frequency for each of their strings $s \in C_r$. Because a second C_{r_2} random array with $n = 4$ will have exactly the values in the average case, there is no reason to take the distance between them, which will be always zero. The average distance is therefore measured between a given classification C and the average equiprobable case denoted by EC such that $d_{pr}(C, EC)$ has meaning. It follows then that if $d(C_1, C_2) < PR$, C_1 and C_2 will be correlated. On section 1.4 we present the distance results for experiment.

Unlike the raw distance, this distance takes into consideration the string's frequency, which allows it to look more deeply into the possible correlation between the classification arrays. However it should be noticed that for most cases the strings composed by a sequence of zeros (or ones) represent more than half of the total frequencies for all cases, so the rest is divided by the other $2^n - 1$ strings. The result is that the average probability distance between two random arrays will be unbalanced and unfair compared to the average distance, which will always assign an equiprobable value to all the strings. One solution would be to dismiss the string $(0)^n$ which always come first, and therefore exerts the strongest attraction effect and biases the distance, but we decided to keep the experiment as far as possible from any manipulation. Indeed it will be seen that, despite these possible biases, a correlation between all of the systems did emerge.

1.4. Results

This is an example of two classification arrays for images (IMG), for $n = 5$ (left table) and $n = 6$ (right table in two columns).

n=5	n=6	
00000	000000	011011
01111	011111	010111
00111	001111	001011
00100	000111	010011
01000	010000	011101
01100	001000	011001
01110	011000	010010
01101	001100	010100
01011	011110	011010
01010	011100	010101

Table 1.4.0
Example of two classification arrays for images (IMG), for $n = 5$ (left table) and $n = 6$ (right table in two columns). A shifting pattern is clearly distinguishable.

The following tables provide all distance results for $n = 4, 5, 6, 10$ for Turing machines (TM and TMR), cellular automata (ECA and ECAR), images (IMG) and DNA (both possible encodings DNA1 and DNA2):

The results within the tables can be understood as follows: from the fact that the distances between most of the systems is smaller than the average distance along the border of each table it follows that there is a correlation between all of them, albeit to a greater or lesser degree.

Something to take into account is the mentioned fact that some abstract systems failed to generate certain strings when n grew large and when the systems were fed by regular inputs. The cases in which that happened

D1, n=4	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	8.	6.	6.	8.	8.	8.	11.7
TMR	8.	0.	2.	10.	10.	8.	0.	11.7
ECA	6.	2.	0.	8.	10.	8.	2.	11.7
ECAR	6.	10.	8.	0.	10.	12.	10.	11.7
IMG	8.	10.	10.	10.	0.	2.	10.	11.7
DNA1	8.	8.	8.	12.	2.	0.	8.	11.7
DNA2	8.	0.	2.	10.	10.	8.	0.	11.7
AVG	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7

1.4.1 Table with all raw distances between all systems for strings of length four.

D1, n=5	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	20.	22.	28.	22.	24.	22.	33.24
TMR	20.	0.	20.	22.	24.	20.	16.	33.24
ECA	22.	20.	0.	14.	26.	22.	14.	33.24
ECAR	28.	22.	14.	0.	26.	24.	16.	33.24
IMG	22.	24.	26.	26.	0.	4.	32.	33.24
DNA1	24.	20.	22.	24.	4.	0.	28.	33.24
DNA2	22.	16.	14.	16.	32.	28.	0.	33.24
AVG	33.24	33.24	33.24	33.24	33.24	33.24	33.24	33.24

1.4.2 Table with all raw distances between all systems for strings of length five.

include TMR for $n = 6$ and $n = 10$, ECA and ECAR for $n = 10$. Table 1.4.5 summarizes these facts.

A solution to this problem that we will address in the future is to take a bigger set of automata (or all the complete automata unlike the last step) in order to produce more strings, and therefore assure the requisite variety. In the meantime, to overcome the problem we attached at the tail of each classification the missing strings in a (pseudo-) random order assigning them a zero frequency.

1.5. Outcome interpretation

There are some important remarks to be made. We certainly expected a correlation between the outcome of abstract systems with the same type of input such as regular on the one hand and pseudo-random on the other. However it can be seen that the distances between Turing machines and

D1, n=6	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	94.	82.	96.	90.	108.	88.	132.25
TMR	94.	0.	76.	76.	72.	66.	116.	132.25
ECA	82.	76.	0.	80.	120.	100.	72.	132.25
ECAR	96.	76.	80.	0.	92.	88.	98.	132.25
IMG	90.	72.	120.	92.	0.	38.	142.	132.25
DNA1	108.	66.	100.	88.	38.	0.	122.	132.25
DNA2	88.	116.	72.	98.	142.	122.	0.	132.25
AVG	132.25	132.25	132.25	132.25	132.25	132.25	132.25	132.25

1.4.3 Table with all raw distances between all systems for strings of length six.

D1, n=10	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	23068.	22250.	24092.	22558.	23028.	23180.	24645.1
TMR	23068.	0.	21616.	17832.	17546.	19050.	17152.	24645.1
ECA	22250.	21616.	0.	24806.	21602.	20524.	20346.	24645.1
ECAR	24092.	17832.	24806.	0.	21984.	21086.	21266.	24645.1
IMG	22558.	17546.	21602.	21984.	0.	14350.	25100.	24645.1
DNA1	23028.	19050.	20524.	21086.	14350.	0.	17910.	24645.1
DNA2	23180.	17152.	20346.	21266.	25100.	17910.	0.	24645.1
AVG	24645.1	24645.1	24645.1	24645.1	24645.1	24645.1	24645.1	24645.1

1.4.4 Table with all raw distances between all systems for strings of length ten.

n	2^n	Burnside[2^n]	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2
4	16	6	6	6	6	6	6	6	6
5	32	10	9	10	10	10	10	10	10
6	64	20	16	20	20	20	20	20	20
10	1024	272	45	272	256	116	272	272	272

1.4.5 Table providing the number of strings generated by each system compared to what should be expected after applying Burnside's lemma to the total 2^n possible strings.

Elementary Cellular Automata with regular inputs versus Turing machines with pseudo-random inputs were not either similar or different enough to take position on that regard yet.

As a consequence of the experiment development we found that the average distance between sets of 10 images turned out to be surprisingly

D2, n=4	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	4.41369	3.41342	5.2972	0.909266	6.86909	8.92988	9.85715
TMR	4.41369	0.	2.62537	1.27669	4.40285	2.51439	4.51619	9.85715
ECA	3.41342	2.62537	0.	2.63625	3.5034	5.13976	5.93712	9.85715
ECAR	5.2972	1.27669	2.63625	0.	5.28636	2.50351	3.63268	9.85715
IMG	0.909266	4.40285	3.5034	5.28636	0.	6.85825	8.91904	9.85715
DNA1	6.86909	2.51439	5.13976	2.50351	6.85825	0.	2.38888	9.85715
DNA2	8.92988	4.51619	5.93712	3.63268	8.91904	2.38888	0.	9.85715
AVG	9.85715	9.85715	9.85715	9.85715	9.85715	9.85715	9.85715	9.85715

1.4.6 Table with all probability distances between all systems for strings of length four.

D2, n=5	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	2.38534	1.80226	3.22191	0.49913	4.03339	4.87336	5.40595
TMR	2.38534	0.	1.44115	1.01446	2.17537	1.64805	2.66591	5.40595
ECA	1.80226	1.44115	0.	2.15161	1.8944	3.08919	3.60881	5.40595
ECAR	3.22191	1.01446	2.15161	0.	3.18983	1.0081	1.65144	5.40595
IMG	0.49913	2.17537	1.8944	3.18983	0.	3.81305	4.84127	5.40595
DNA1	4.03339	1.64805	3.08919	1.0081	3.81305	0.	1.36411	5.40595
DNA2	4.87336	2.66591	3.60881	1.65144	4.84127	1.36411	0.	5.40595
AVG	5.40595	5.40595	5.40595	5.40595	5.40595	5.40595	5.40595	5.40595

1.4.7 Table with all probability distances between all systems for strings of length five.

small. In most cases images differed by less than two percent by frequency distribution of their strings for any n. By instance, for $n = 4$ and $n = 5$ less than two moves were necessary in average to get one classification array from another.

Turning to the case of the DNA, as a physical repository of information submitted to physical laws (and to what is usually taken as noise), we expected a higher degree of correlation with images (IMG) as repositories also submitted to physical constraints. However we did not expect such a strong correlation. Nevertheless, we should remark the discrepancy outcome when taking one or another of the two possible encodings for transforming a DNA fragment into a binary string. It was found that one encoding was strongly correlated to some systems while the other encoding was weaker correlated to them but stronger to others. Something remarkable is the strong correlation of the DNA first encoding to images (IMG), they almost match having

D2, n=6	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	1.296	0.8726	1.675	0.308	2.312	2.567	2.89
TMR	1.296	0.	0.863	0.645	1.131	1.0164	1.363	2.89
ECA	0.872	0.863	0.	1.040	0.912	1.86837	2.119	2.89
ECAR	1.67	0.645	1.040	0.	1.60	0.90449	1.146	2.89
IMG	0.3086	1.131	0.9128	1.60	0.	2.0691	2.49	2.89
DNA1	2.312	1.0164	1.8683	0.90449	2.0691	0.	0.693	2.89
DNA2	2.567	1.363	2.1196	1.1460	2.4949	0.69323	0.	2.89
AVG	2.89	2.89	2.89	2.89	2.89	2.89	2.89	2.89

1.4.8 Table with all probability distances between all systems for strings of length six.

D2, n=10	TM	TMR	ECA	ECAR	IMG	DNA1	DNA2	AVG
TM	0.	0.085	0.056	0.134	0.040	0.166	0.179	0.191
TMR	0.085	0.	0.067	0.095	0.056	0.088	0.103	0.191
ECA	0.056	0.067	0.	0.121	0.058	0.144	0.156	0.191
ECAR	0.134	0.095	0.121	0.	0.123	0.097	0.105	0.191
IMG	0.040	0.056	0.058	0.123	0.	0.131	0.152	0.191
DNA1	0.166	0.088	0.144	0.097	0.131	0.	0.0515	0.191
DNA2	0.179	0.103	0.156	0.105	0.152	0.0515	0.	0.191
AVG	0.191	0.191	0.191	0.191	0.191	0.191	0.191	0.191

1.4.9 Table with all probability distances between all systems for strings of length ten.

the same frequency array, while DNA2 (second encoding) correlation with images was more weakly correlated but stronger to TM, something that we could not explain so far.

In any case, they all showed some degree of correlation and further work taking bigger and different repositories and trying to interpret the results based on asymmetries shall be made. An evocative view concerning the difference between the classifications is that they are due to the computational particularities of each. In other words, each classification difference is a partial encoding of the particular properties of that system.

For some cases, particularly between abstract vs. real-world systems, It turned out that the main -and sometimes the only- difference between the positions of strings within the arrays was the string with the repeated pattern $(01)^n$. The frequency of appearance of a string of the form $(x, not[x])^n$ turned out to be sometimes higher ranked when n grew. According to algorithmic probability the program to describe or produce the string $0101\ldots$ should be shorter and simpler compared to the programs producing more random-looking data which, unlike $(01)^n$, would have little or no evident structure. However, taking as an example the string 0101 one of the possible description programs is actually $(01)^2$ which is not any shorter than the original description so one could expect to find it lower ranked for small n because it is already so short that further compression seems difficult. However, when n grows the complexity value of the string in comparison with its length decreases substantially. For instance, when $n = 10$ the string 0101010101 can be shortened by $(01)^{10}$ which is not substantially shorter but shorter, for this particular encoding, compared to the cases $(01)^n$ when $n = 4, 5, 6$. By contrast, a pattern like $(01)^n$ from black-and-white images could be expected to be lower ranked when n grows due to the fact that any disturbance - namely noise - might easily destroy the pattern decaying into a string of lower algorithmic probability and then closer to the bottom of the classification. It seems clear that a structured string like that can be easily destroyed by changing any digit. By contrast, a non-symmetrical string as 0100 would still be liable to fall into another string of the same complexity even after changing one or two digits. For example, applying two different bit changes to the string 0100 would produce the string 0010 which remains in the same complexity class. By contrast, any one or two (different bit) changes to 0101 will always produce a string of lower algorithmic complexity. Furthermore, if n grows the probability of making the string to decay by any change due to possible noise will be higher. The probability of preserving complexity under a $2 - bit$ transformation in a repeating pattern such as the example above is zero while for the general non-symmetrical case it will be greater than zero. This phenomena can be explained by different approaches, either by measuring the probability of a string as we just did, or by taking the Hamming distance between strings from same complexity classes, which will turn out to be equivalent to the first approach.

If the probability of change of any digit in a string s is uniformly distributed (any digit has the same probability to be changed) it follows that: $Pr[010101 \to 101010] > Pr[000100 \to 001000]$, where the arrow means that the string s_1 becomes s_2. Or by measuring the Hamming distance it turns out that the number of changes that a string needs to undergo in order to remain under the same complexity class can be described as follow: HammingDistance[010101,101010]$=6$, HammingDistance[000100,001000]$=$ 2. In other words, the shortest path for transforming 010101 into 101010 requires 6 changes to preserve its complexity while the string 000100 two to become 001000 a string of the same complexity class. It is clear that performing six precise bit changes are less probable than performing two. Moreover, the only chance for the first string 010101 to remain in the same complexity class is to become that particular string 101010, while for the second string 000100 there are other possibilities: 001000, 110111 and 111011.

These facts seem relevant to explore possible explanations for the differences found between some of the classification arrays in which some kind of noise is involved and in fact expected when taking sources from the real world. What we claim is that such differences could be explained[l] in terms of pure information and that the discrepancy could be consequential to the submission of those systems to physical laws (in particular the second law of thermodynamics).

1.6. Conclusions and possible applications

We think that our method could efficiently resolve the problem of defining and calculating the Kolmogorov-Chaitin complexity for short sequences. Our approach suggests an alternative method to decide, given two strings, which of them has a higher Kolmogorov-Chaitin complexity, in other terms, their relative complexity. And by following the experiment for all the strings shorter or equal to a given string it would be possible to calculate its absolute Kolmogorov-Chaitin complexity. It would suffice to perform the procedure and make the comparison with the outcome. It is important to bear in mind that the comparisons were made after applying the Burnside's lemma, when the lemma is avoided the distance between the respective distributions is even shorter since at it has been shown, the distributions group the strings by symmetries that preserve their complexity.

An outcome of our experiment[m] is that the classifications we built from different data sources seem to be strongly correlated. We found that all or most frequency arrays were correlated within a range that goes from

[l]We have much more to say on this issue but unfortunately not enough space for. We think in fact that abstract systems would produce such noise after letting them run longer time and interact between them. It is clear that accepting an essential difference between randomness and pseudo- randomness would yield to the conclusion that there are two different and incompatible sources. But we think that our results suggest the contrary.

[m]A website with all and the complete results of the whole experiment is available at http://www.mathrix.org/experimentalAIT/

weakly to strongly correlated. For the case of real-world information repositories they seem to converge towards the arrays of abstract systems fed by (pseudo) random inputs when n grows. Our research suggests that this linkage between classification arrays from several abstract and real-world systems is due to a shared distribution predicted by the algorithmic probability and the Kolmogorov-Chaitin complexity.

Moreover, and in opposition to the widespread believe, our work is an example of an empirical application of algorithmic information theory. Moreover, we have found that the frequency distribution from several real-world data sources also approximates the same distribution, suggesting that they probably come from the same kind of computation, supporting current contemporary claims about nature as performing computations.[8,11] Our outcome might also be of utility as other phenomenological laws, such as Benford's and Zipf's laws, which are currently used for detecting frauds in tax refunds or to certify the reliability of data. Moreover, if such hypotheses can be either verified or refuted many interesting consequences for several domains can be enriched, including discussions about nature as performing computations,[8,11] whether metaphorical or otherwise, bringing it back to the mathematical territory. A paper with mathematical formulations and precise conjectures is currently being prepared to be published.

References

1. G.J. Chaitin, *Algorithmic Information Theory*, Cambridge University Press, 1987.
2. G.J. Chaitin, *Information, Randomness and Incompleteness*, World Scientific, 1987.
3. G.J. Chaitin, *Meta-Math! The Quest for Omega*, Pantheon Books NY, 2005.
4. C.S. Calude, *Information and Randomness: An Algorithmic Perspective (Texts in Theoretical Computer Science. An EATCS Series)*, Springer; 2nd. edition, 2002.
5. Kirchherr, W., M. Li, and P. Vitányi. The miraculous universal distribution. *Math. Intelligencer* 19(4), 7-15, 1997.
6. M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, 1997.
7. A.K.Zvonkin, L. A. Levin. "The Complexity of finite objects and the Algorithmic Concepts of Information and Randomness.", *UMN = Russian Math. Surveys*, 25(6):83-124, 1970.
8. S. Lloyd, *Programming the Universe*, Knopf, 2006.
9. R. Solomonoff, The Discovery of Algorithmic Probability, *Journal of Computer and System Sciences*, Vol. 55, No. 1, pp. 73-88, August 1997.
10. R. Solomonoff, *A Preliminary Report on a General Theory of Inductive In-*

ference, (Revision of Report V-131), Contract AF 49(639)-376, Report ZTB-138, Zator Co., Cambridge, Mass., Nov, 1960
11. S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.